IMAGE PROCESSING & ANALYSIS

IMAGE PROCESSING & ANALYSIS

Image enhancement

Point operations

Local operations

Noise reduction / Image smoothing

Edge & Line detection

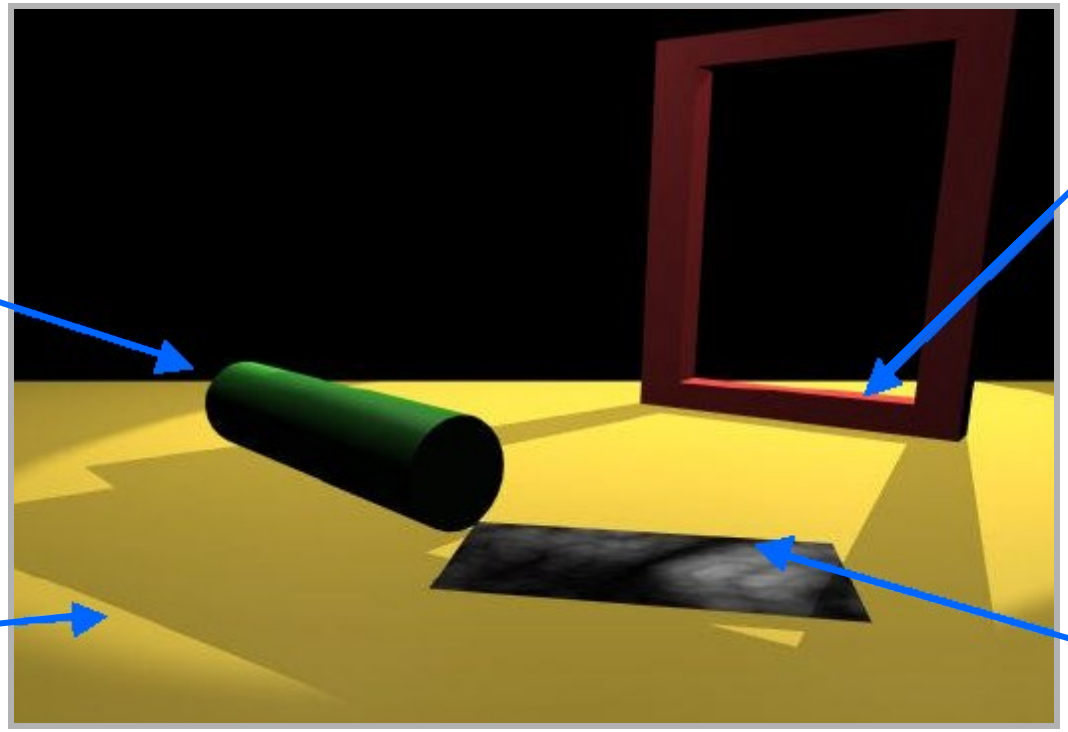
Edge detection

First order & second order detectors

Edge Detection

- What's an edge?
  - "He was sitting on the edge of his seat."
  - "I almost ran off the edge of the road."
  - "She was standing by the edge of the woods."
  - "Film negatives should only be handled by their edges."
  - "She paints with a hard edge."
  - "We are on the edge of tomorrow."
  - "He likes to live life on the edge."
  - "She is feeling rather edgy."
- The definition of **edge** is not always clear.
- In **Computer Vision**, **edge** is usually related to a discontinuity within a local set of pixels.

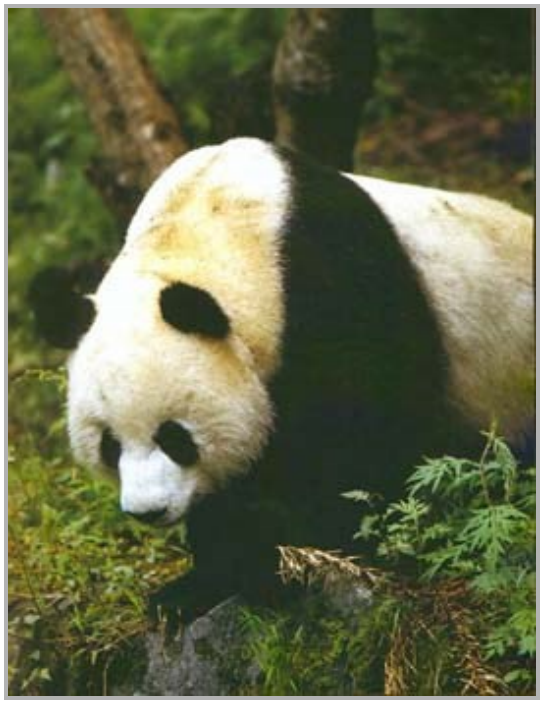
Discontinuities



- A: Depth discontinuity: abrupt depth change in the world
- B: Surface normal discontinuity: change in surface orientation
- C: Illumination discontinuity: shadows, lighting changes
- D: Reflectance discontinuity: surface properties, markings

Goal

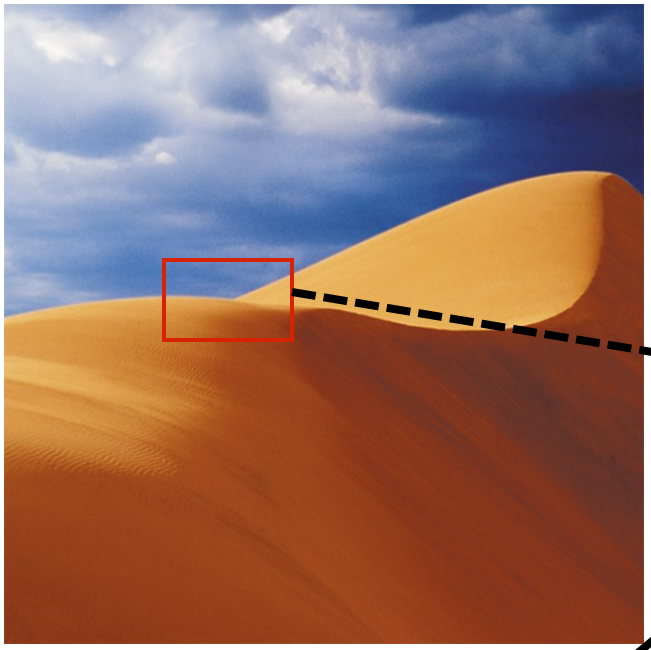
- Devise computational algorithms for the extraction of **significant edges** from the image.
- What is meant by significant is unclear.
  - Partly defined by the context in which the edge detector is being applied



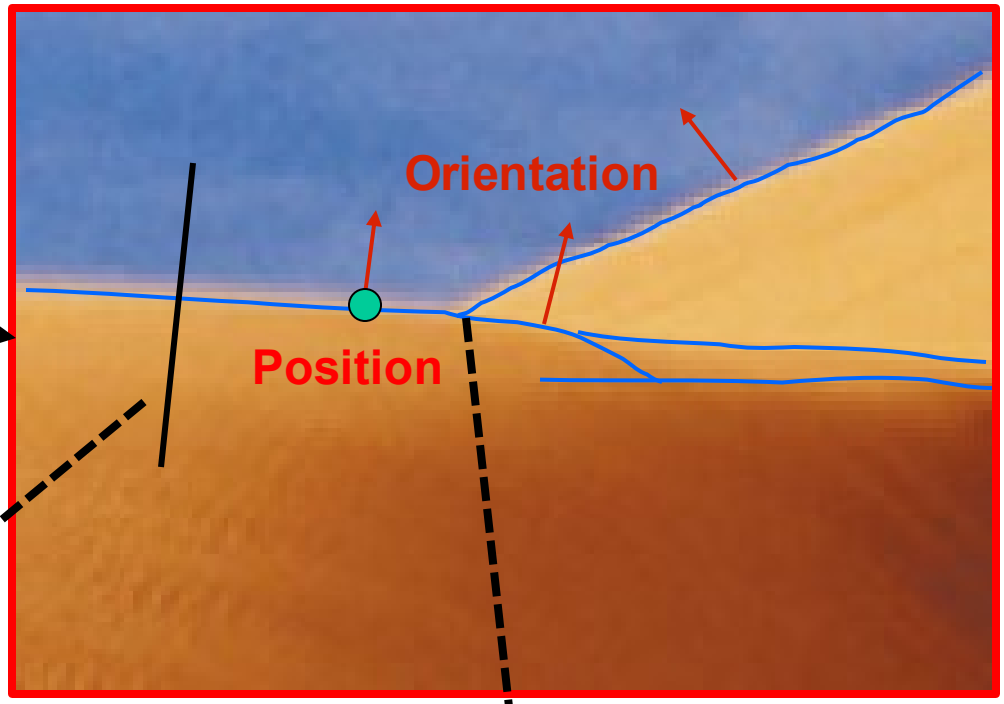
Edgels

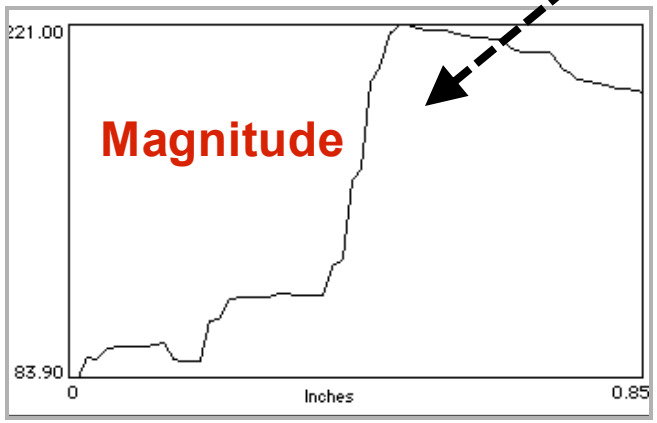
- Define a local edge or edgel to be a rapid change in the image function over a small area
  - implies that edgels should be detectable over a local neighborhood
- **Edgels** are **NOT contours**, **boundaries**, or **lines**
  - edgels may lend support to the existence of those structures
  - these structures are typically constructed from edgels
- Edgels have properties
  - Position
  - Orientation
  - Magnitude

Properties of an edge



zoom



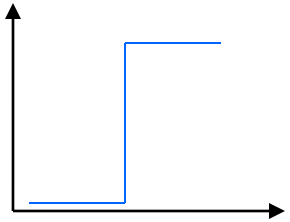


zoom

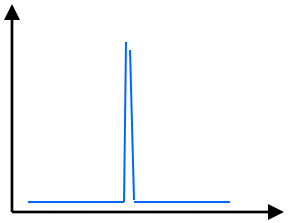
Locating edgels

Rapid change in image => high local gradient => differentiation

f(x) - step edge

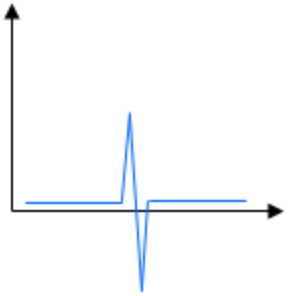


1st Derivative → f ’(x)



maximum

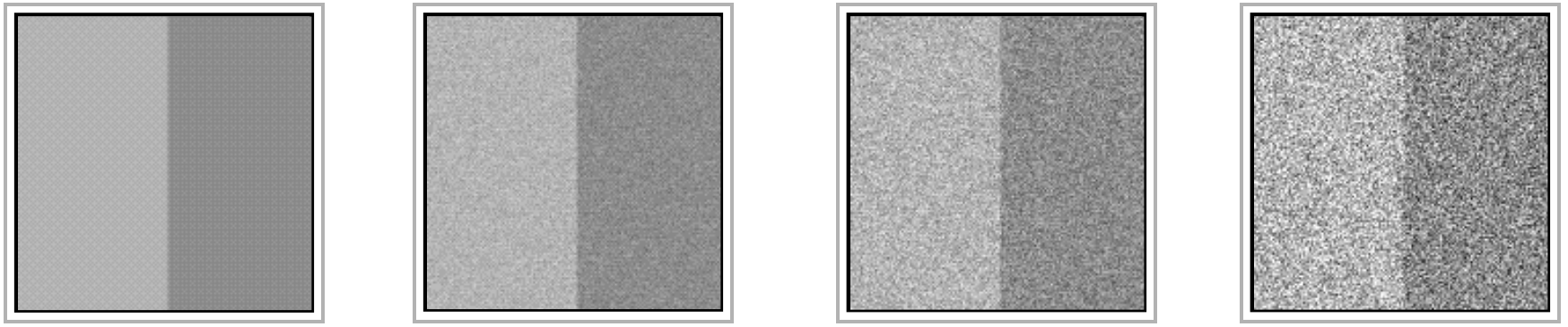
2nd Derivative → f ”(x)



zero crossing

Edge degradation by noise

Increasing noise



Ideal step edge

Step edge + noise

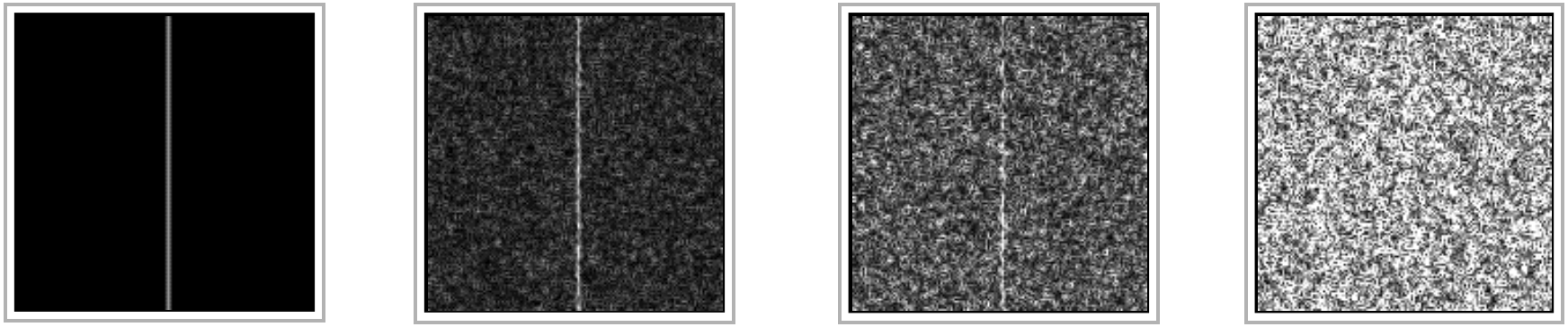


image gradient

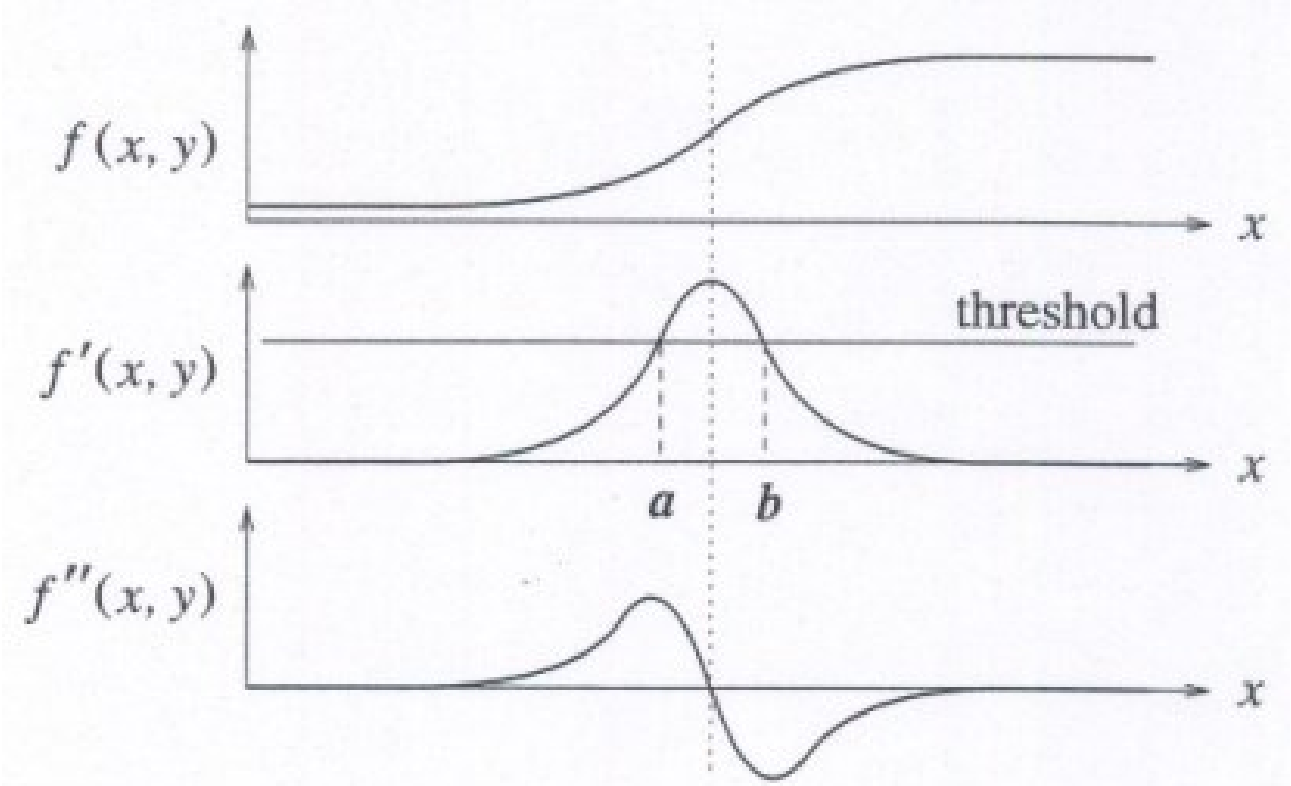
Real Image

Edge Detection: typical steps

- Noise Smoothing
  - Suppress as much noise as possible while retaining 'true' edges
  - In the absence of other information, assume noise with a Gaussian distribution
- Edge Enhancement
  - Design a filter that responds to edges; high filter output at edge pixels and low elsewhere
- Edge Localization
  - Determine which edge pixels should be discarded as noise and which should be retained
    - thin wide edges to 1-pixel width (nonmaximum suppression)
    - establish minimum value to declare a local maximum from edge filter to be an edge (thresholding)

Edge Detection: typical steps

- Edge Localization (cont.)



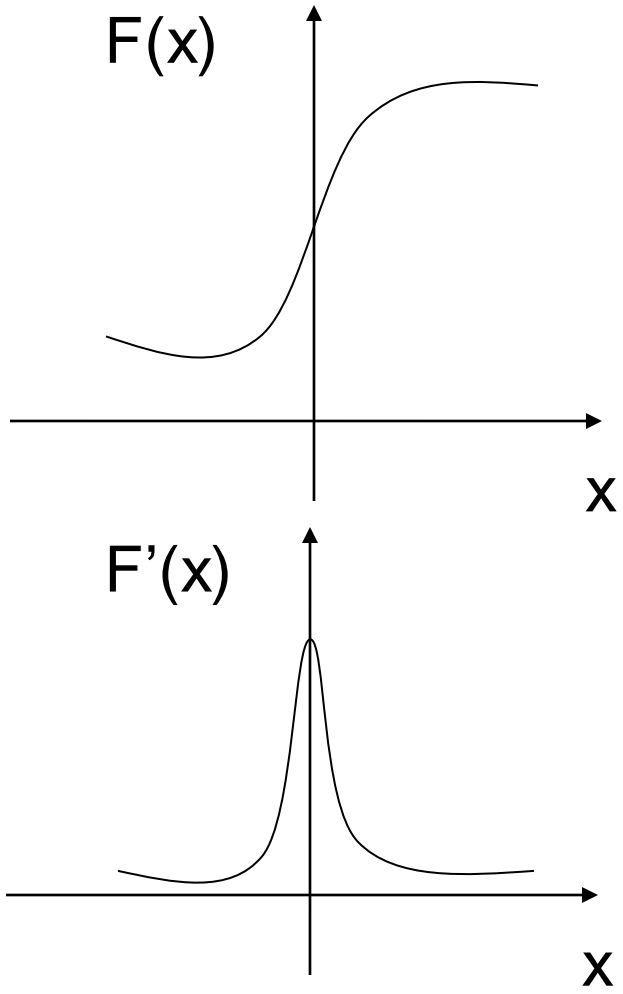
If a threshold is used for detection of edges, all points between a and b will be marked as edge pixels.
However, by removing points that are not a local maximum in the first derivative,
edges can be detected more accurately.
This local maximum in the first derivative corresponds to a zero crossing in the second derivative.

source: "Machine Vision" book, R.Jain et al

Edge Detection Methods

- 1st Derivative estimate
  - Gradient edge detection
    - Prewitt & Sobel detectors
  - Compass edge detection
  - Canny edge detector
- 2nd Derivative estimate
  - Laplacian
  - Difference of Gaussians
- Parametric Edge Models

Gradient Methods



Edge= sharp variation

Large first derivative

Gradient of a Function

- Assume f is a continuous function in (x,y). Then

$$\Delta_x = \frac{\partial f}{\partial x}, \quad \Delta_y = \frac{\partial f}{\partial y}$$

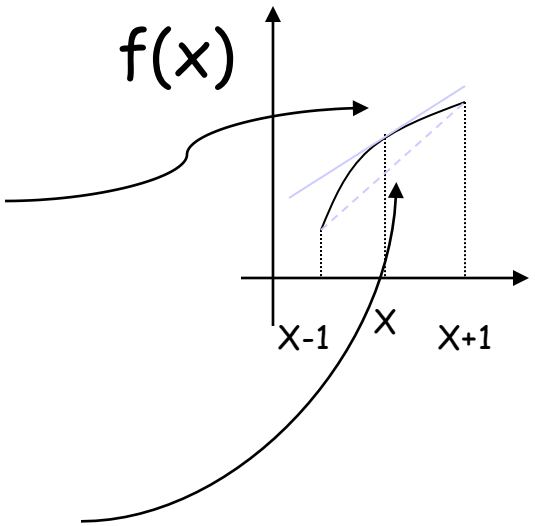
- are the rates of change of the function f in the x and y directions, respectively.
- The vector (Δ_x, Δ_y) is called the gradient of f.
- This vector has a magnitude: $s = \sqrt{\Delta_x^2 + \Delta_y^2}$

and an orientation: $\theta = \tan^{-1}\left(\frac{\Delta_y}{\Delta_x}\right)$

- θ is the direction of the maximum change in f.
- S is the size of that change.

Digital approximation

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$



Convolve with:

-1	1

$$\frac{df(x)}{dx} \cong \frac{f(x+1) - f(x-1)}{2}$$

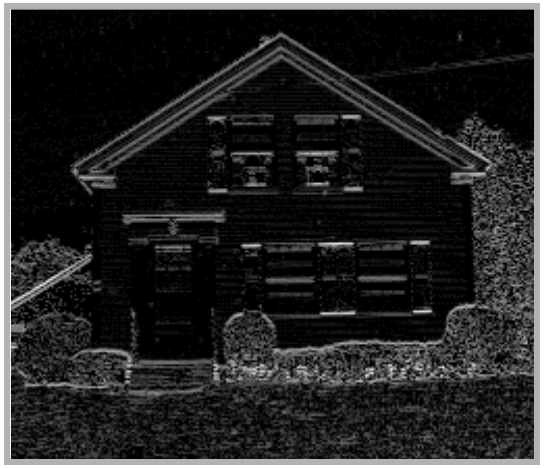
Convolve with:

-1	0	1

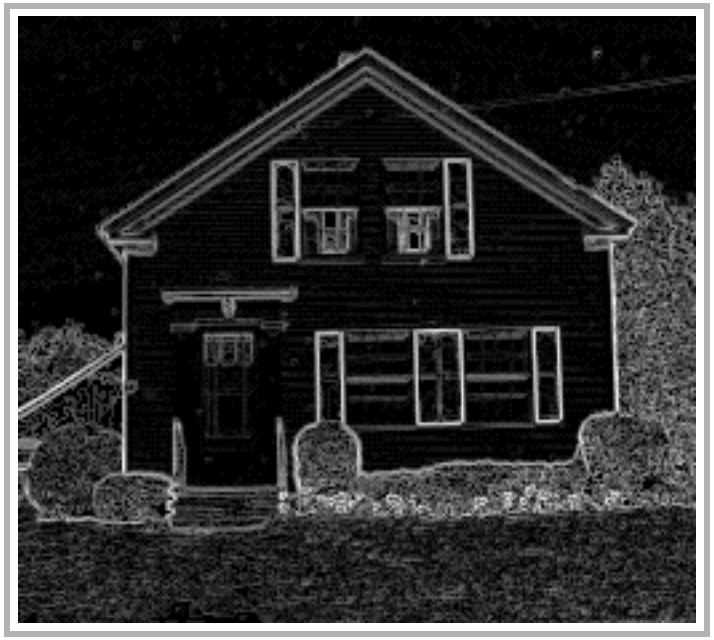
1x2 Example

1x2 Vertical

1x2 Horizontal



Combined



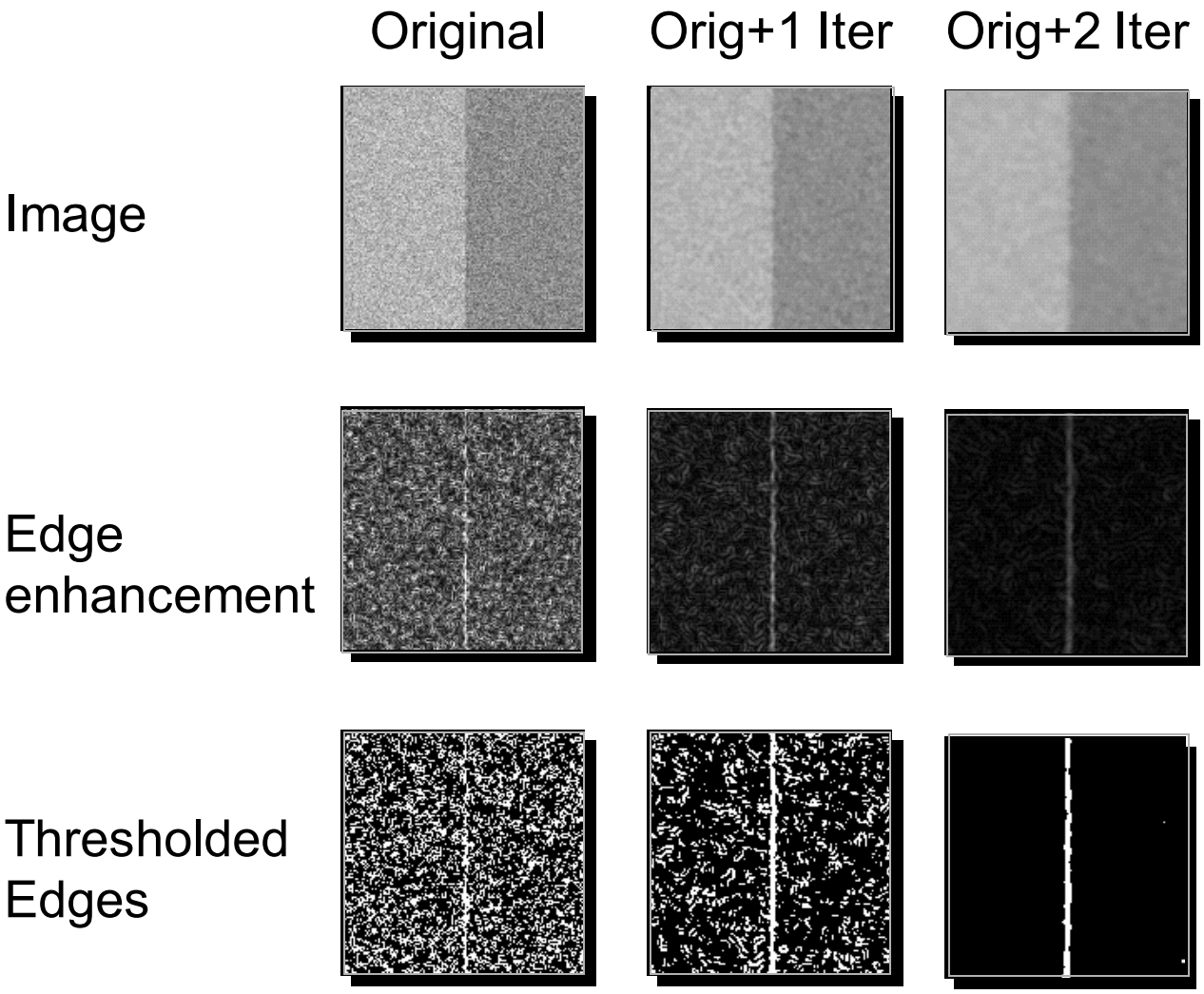
Smoothing and Edge Detection

- Derivatives are 'noisy' operations
  - edges are a high spatial frequency phenomenon
  - edge detectors are **sensitive to** and **emphasize** noise
- Averaging reduces noise
  - spatial averages can be computed using masks

$1/9 \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ $1/16 \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

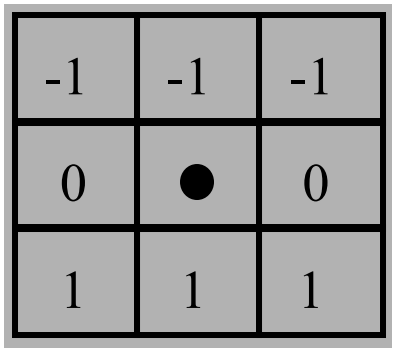
- Combine smoothing with edge detection.

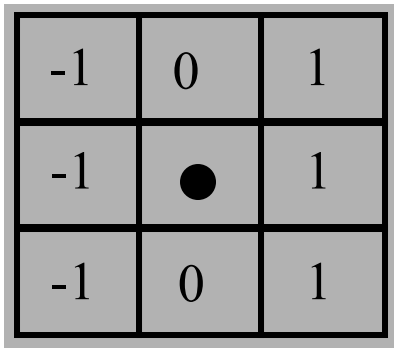
Effect of Blurring

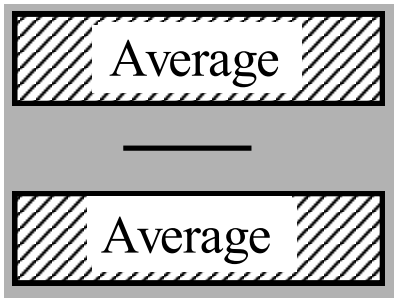


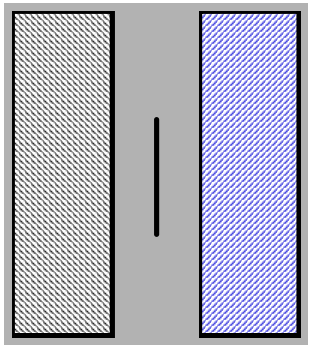
Combining Smoothing and Edge Detection

- Applying simple averaging before differencing is equivalent to taking the difference of averages on either side of the central pixel.









Prewitt Operator

$P_1 =$

-1	-1	-1
0	0	0
1	1	1

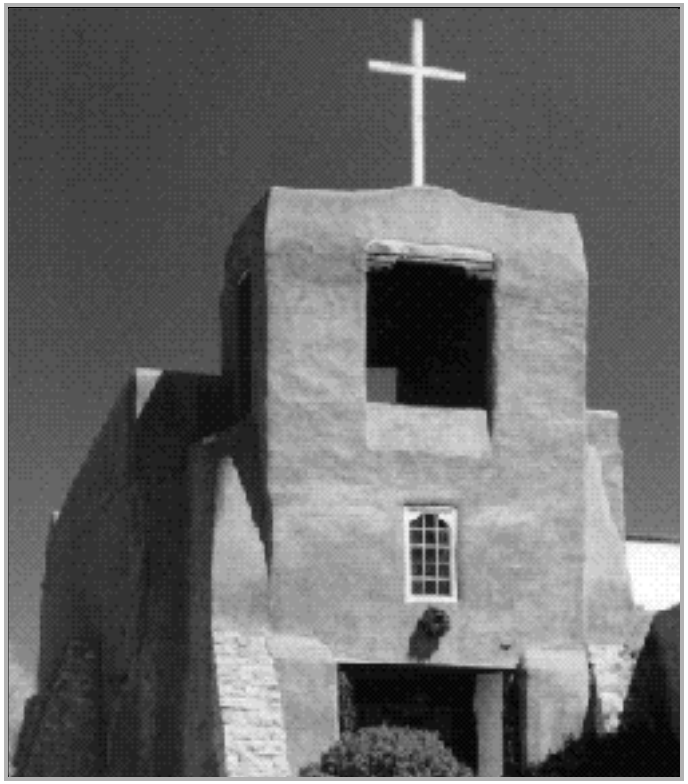
$P_2 =$

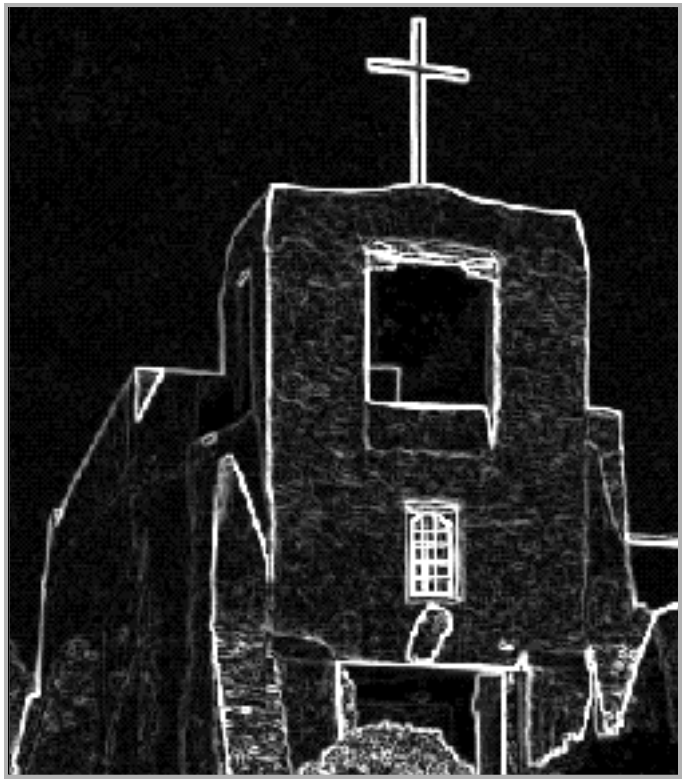
-1	0	1
-1	0	1
-1	0	1

$$\text{Edge Magnitude} = \sqrt{P_1^2 + P_2^2}$$

$$\text{Edge Direction} = \tan^{-1}\left(\frac{P_1}{P_2}\right)$$

Prewitt Example





Prewitt

horizontal and vertical edges combined

Sobel Operator

Kernels

$$S_1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad S_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{Gradient/Edge Magnitude} = \sqrt{S_1^2 + S_2^2}$$

$$\text{Gradient Direction} = \tan^{-1}\left(\frac{S_1}{S_2}\right)$$

NOTES:

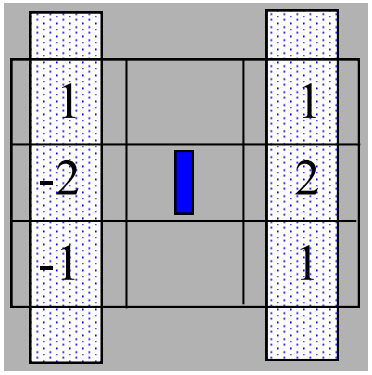
1) S_1, S_2 represent responses to the kernels (above), not the kernel themselves
2) Gradient direction is perpendicular to edge direction

Anatomy of the Sobel

$$\frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = 1/4 * [-1 \;\; 0 \;\; -1] \circledast \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = 1/4 * [\,1 \;\; 2 \;\; 1] \circledast \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Sobel kernel is separable!

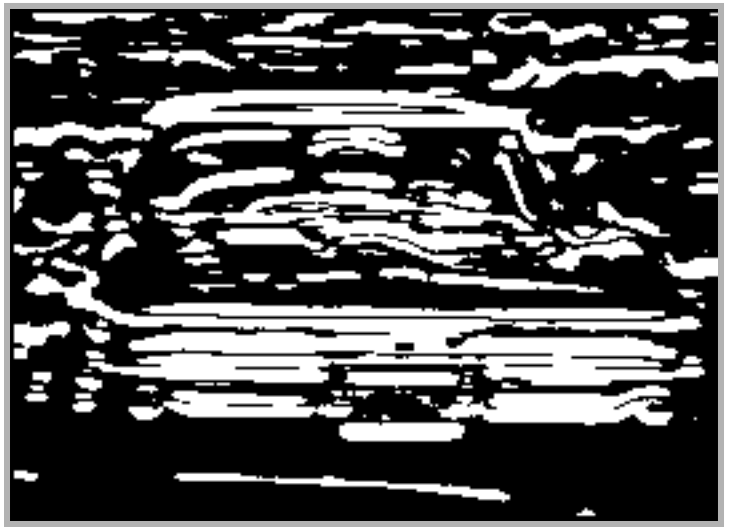


Averaging done parallel to edge

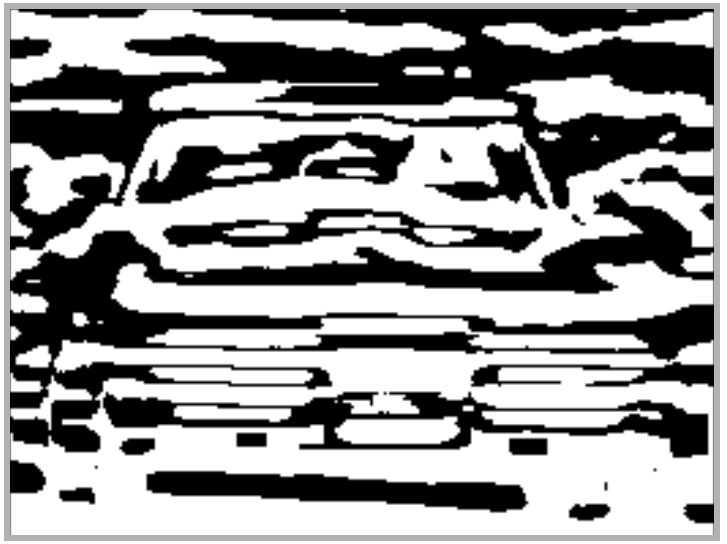
Large Kernels

The Intel IPP library,
gives the 5x5 horizontal kernel as

```
1  2  0  -2 -1
4  8  0  -8 -4
6 12  0 -12 -6
4  8  0  -8 -4
1  2  0  -2 -1
```

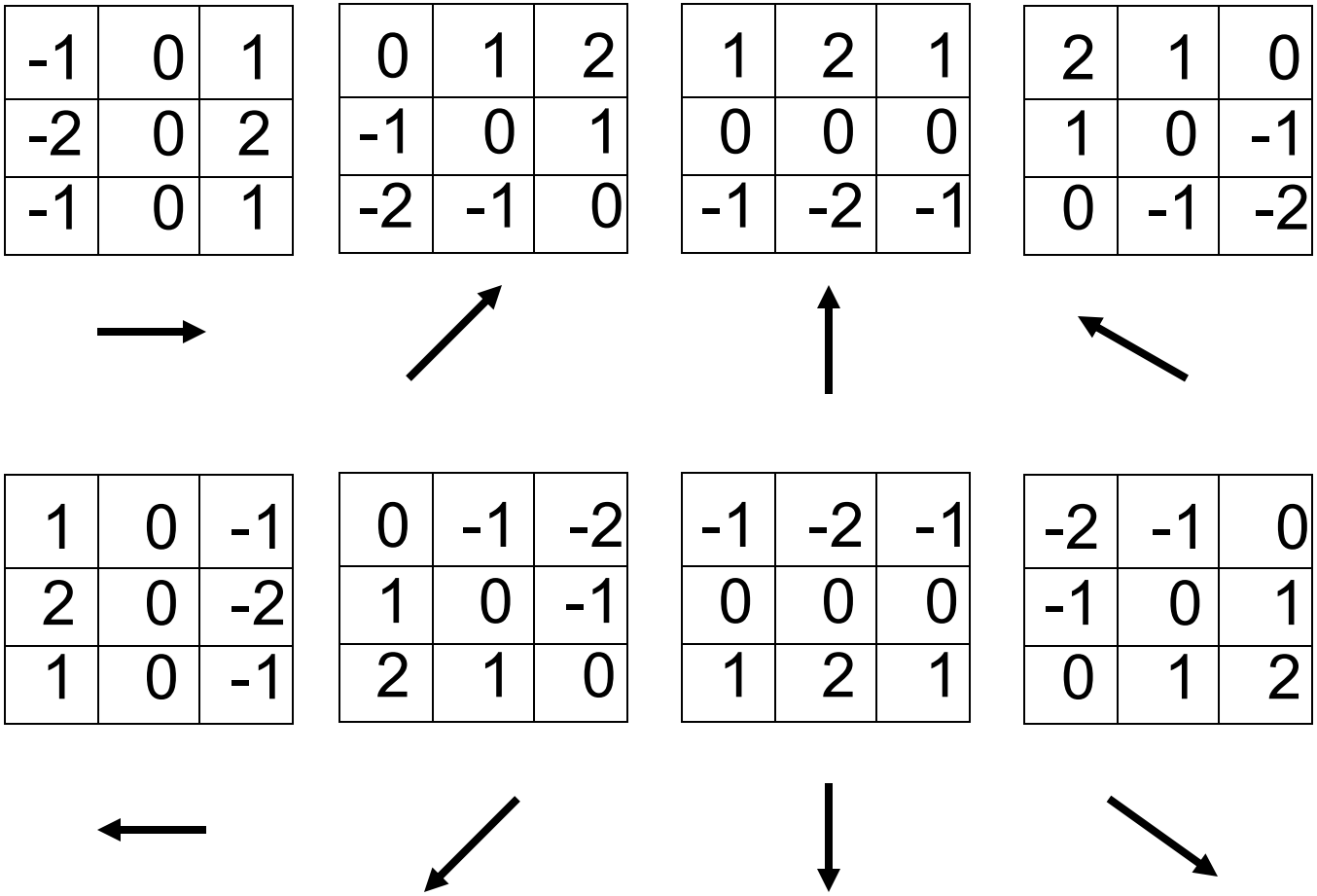


7x7 horizontal edges (only)



13x13 horizontal edges (only)

Robinson Compass Masks

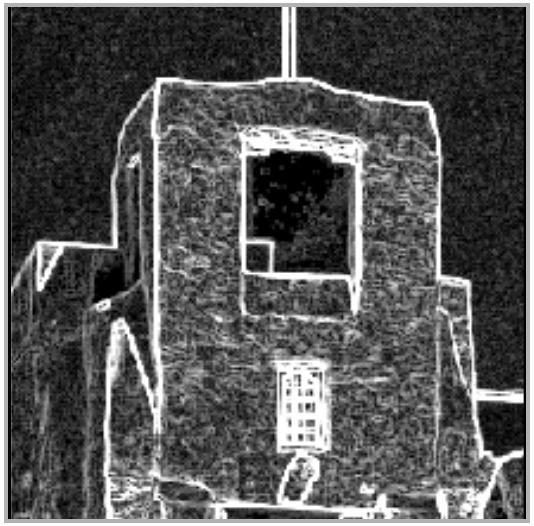


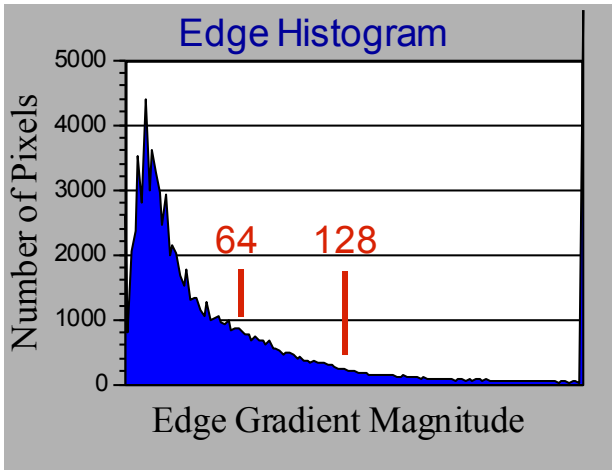
- Select largest response (magnitude)
- Orientation is
  the direction associated with the largest response

Edge Thresholding

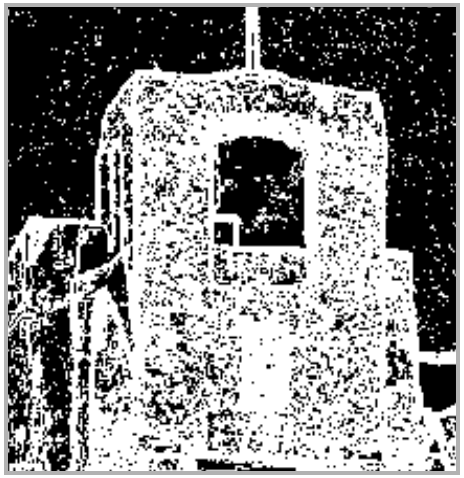
- Global approach

edge magnitude

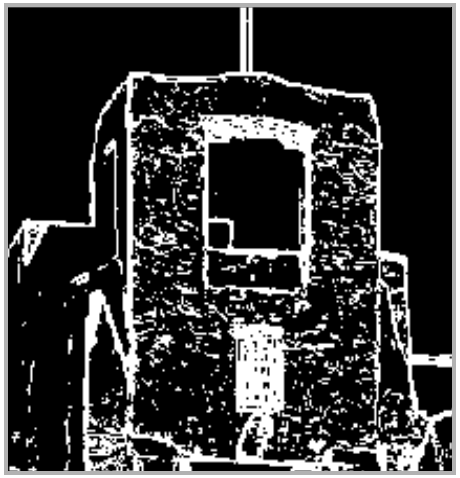




thresholding results



T=64



T=128

Canny Edge Detector

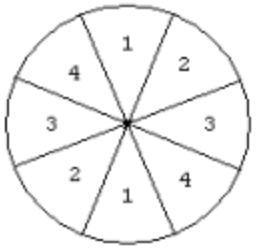
- **Edge detection** involves 3 **steps**:
  - **Noise smoothing**
  - **Edge enhancement**
  - **Edge localization**
- J. Canny formalized these steps to design an optimal edge detector
  - J.F. Canny, "A computational approach to edge detection", IEEE Trans. Pattern Anal. Machine Intelligence (PAMI), vol. PAMI vii-g, pp. 679-697, 1986.
- based on a set of criteria that should be satisfied by an edge detector:
  - **Good detection**. There should be a minimum number of false negatives and false positives.
  - **Good localization**. The edge location must be reported as close as possible to the correct position.
  - **Single response**. Only one response to a single edge.
- Probably, most widely used edge detector

Canny Edge Detector

- **Stage 1. Image Smoothing**
  - The image data is smoothed by a Gaussian function of width specified by the user parameter (σ).
- **Stage 2. Differentiation**
  - The smoothed image, retrieved at Stage 1, is differentiated with respect to the directions x and y.
  - From the computed gradient values x and y, the magnitude and the angle of the gradient can be calculated

Canny Edge Detector

- **Stage 3. Non-Maximum Suppression**
  - The edges can be located at the points of local maximum gradient magnitude.
  - It is done via suppression of non-maxima, that is, points, whose gradient magnitudes are not local maxima.
  - Non-maxima perpendicular to the edge direction, rather than those in the edge direction, have to be suppressed,
  - The algorithm starts off by reducing the angle of gradient to one of the four sectors shown in Figure.
  - The algorithm passes the 3x3 neighborhood across the magnitude array. At each point the center element of the neighborhood is compared with its two neighbors along line of the gradient given by the sector value.
  - If the central value is non-maximum, that is, not greater than the neighbors, it is suppressed.



Canny Edge Detector

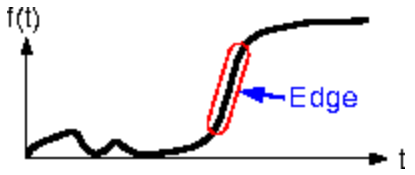
- **Stage 4. Edge Thresholding**
  - The Canny operator uses the so-called "hysteresis" thresholding.
  - Most thresholders use a single threshold limit, which means that if the edge values fluctuate above and below this value, the line appears broken. This phenomenon is commonly referred to as "streaking".
  - Hysteresis counters streaking by setting an upper and lower edge value limit.
  - Considering a line segment,
    - If a value lies above the upper threshold limit it is immediately accepted.
    - If the value lies below the low threshold it is immediately rejected.
    - Points which lie between the two limits are accepted if they are connected to pixels which exhibit strong response.
  - The likelihood of streaking is reduced drastically since the line segment points must fluctuate above the upper limit and below the lower limit for streaking to occur.
  - J. Canny recommends in [Canny86] the ratio of high to low limit to be in the range of two or three to one, based on predicted signal-to-noise ratios.

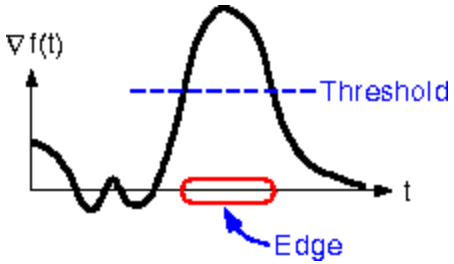
Canny Edge Detector

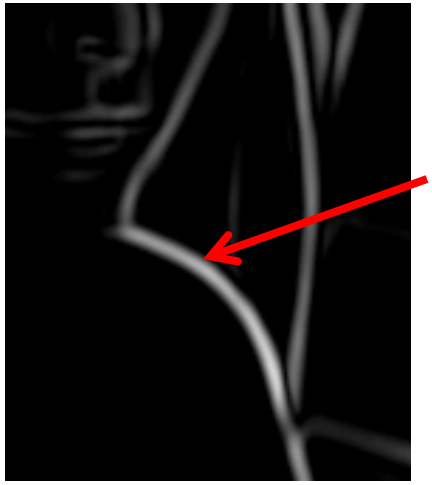
Original image

Gradient magnitude

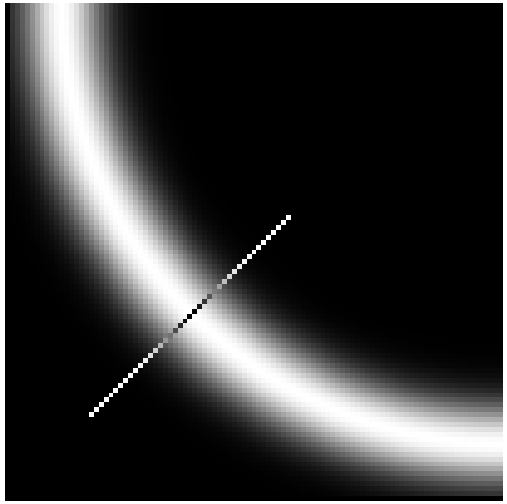
Thresholding







How to turn these thick regions of the gradient into curves?



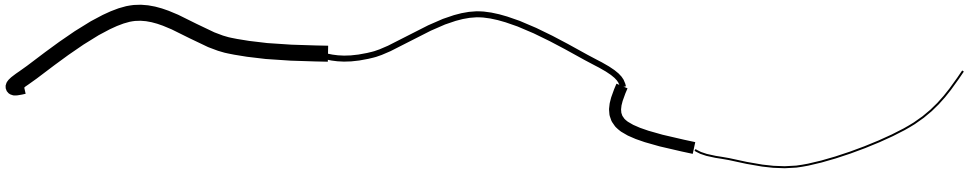
Non-maximum supression

Hysteresis thresholding

Problem: pixels along this edge didn't survive the thresholding

Hysteresis thresholding

- Use a high threshold to start edge curves, and a low threshold to continue them.



original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Canny Results

σ=1, T2=255, T1=1

- Hysteresis threshold method allows
  to add weaker edges (those above *T1)*
  if they are neighbors of stronger edges (those above *T2).*

Canny Results

σ=1, T2=255, T1=220

σ=1, T2=128, T1=1

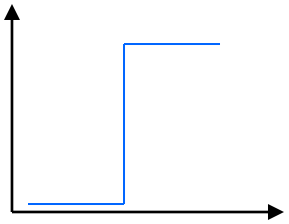
σ=2, T2=128, T1=1

M. Heath, S. Sarkar, T. Sanocki, and K.W. Bowyer, "A Robust Visual Method for Assessing the Relative Performance of Edge-Detection Algorithms" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 12, December 1997, pp. 1338-1359.

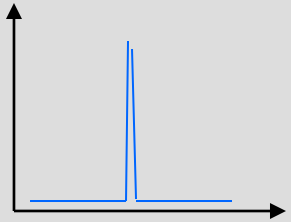
Edges from Second Derivatives

- Digital gradient operators estimate the first derivative of the image function in two or more directions.

f(x) = step edge



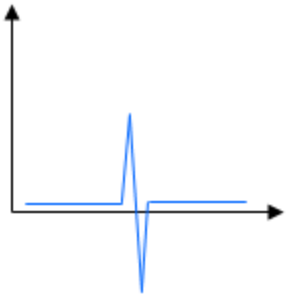
1st Derivative f'(x)



maximum

GRADIENT METHODS

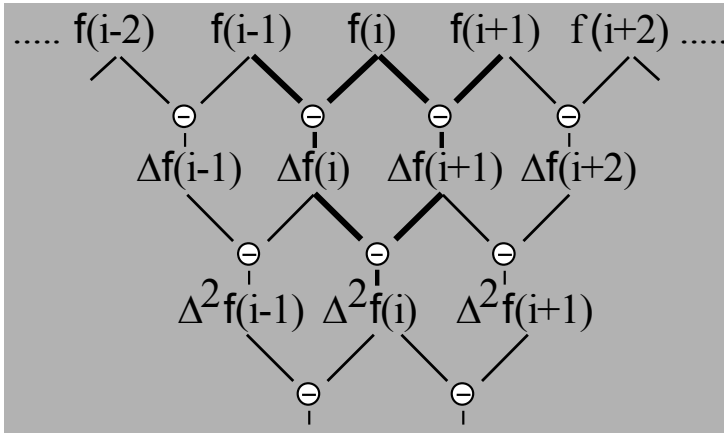
2nd Derivative f''(x)



zero crossing

Second Derivatives

- Second derivative = rate of change of 1st derivative.
- Maxima of first derivative = zero crossings of 2nd derivative.
- For a discrete function,
  derivatives can be approximated by differencing.
- Consider the one dimensional case:



$$\Delta^2 f(i) = \Delta f(i+1) - \Delta f(i)$$

$$= f(i+1) - 2 f(i) + f(i-1)$$

Mask:

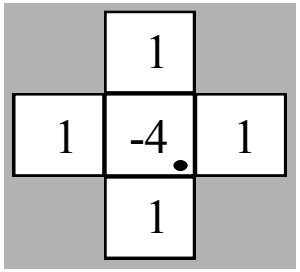
1	-2	1

Laplacian Operator

- Now consider a two-dimensional function f(x,y).
- Can be shown that the smallest possible isotropic second derivative operator is the Laplacian:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Two-dimensional discrete possible approximations:



1	1	1
1	-8	1
1	1	1

-1	2	-1
2	-4	2
-1	2	-1

Note: the OpenCV function Laplacian() uses, by default, the kernel on the left

Example Laplacian Kernels

-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	24	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

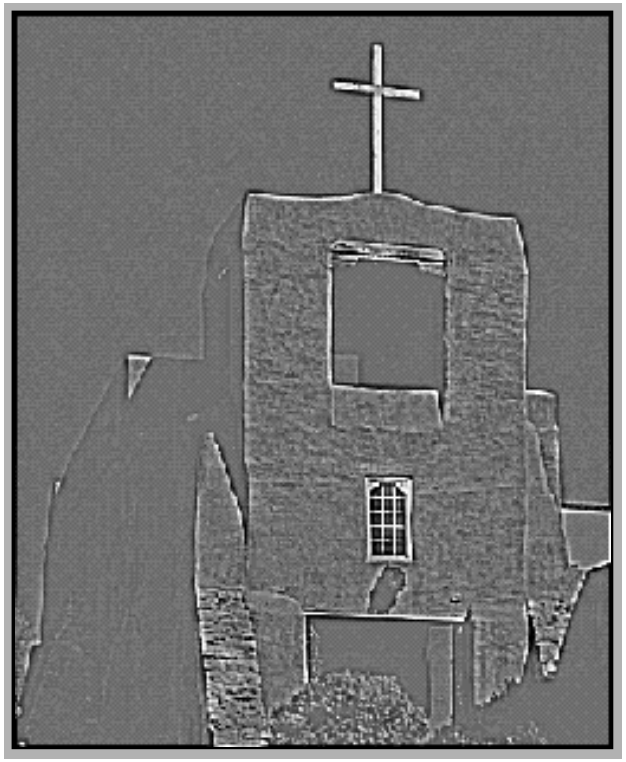
5X5

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	+8	+8	+8	-1	-1	-1
-1	-1	-1	+8	+8	+8	-1	-1	-1
-1	-1	-1	+8	+8	+8	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

9X9

- Note that these are <u>not</u> the <u>optimal</u> approximations to the Laplacian of the sizes shown.

Example Application



5x5 Laplacian filter

9x9 Laplacian filter

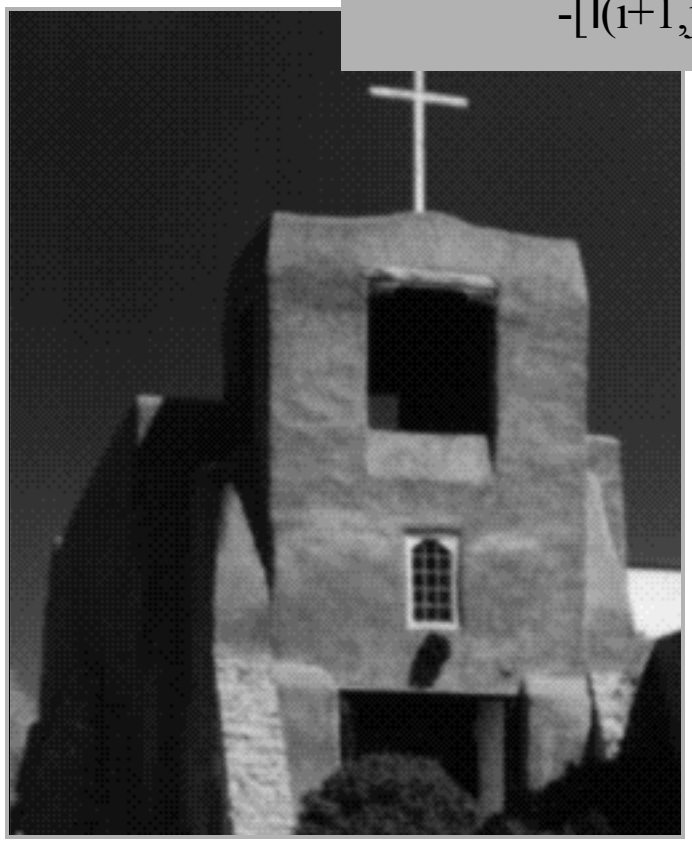
- Edges are located at **zero crossings.**
- To avoid detection of insignificant edges,
  only the zero crossings whose corresponding first derivative is above some threshold should be selected as edge points.
- Note: **for display** purpose, **zero** is represented as a **mid-level gray.**

Enhancement using the Laplacian

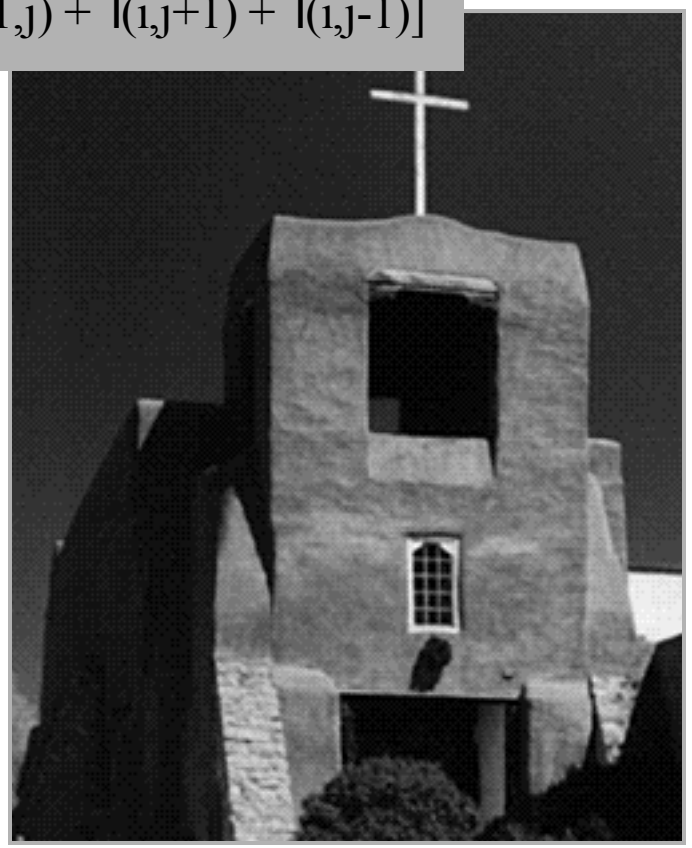
$$I(i,j) - \nabla^2 I(i,j) =$$

$$5\, I(i,j)$$

$$-[I(i+1,j) + I(i-1,j) + I(i,j+1) + I(i,j-1)]$$



Blurred original



3x3 Laplacian enhanced

- Effect is one of deblurring the image

$\nabla^2 G$ Filter

- Marr and Hildreth approach:
  1. Apply Gaussian smoothing using σ's of increasing size:

$$G \circledast I$$

  2. Take the Laplacian of the resulting images:

$$\nabla^2 (G \circledast I)$$

  3. Look for zero crossings.

- Second expression can be written as: $(\nabla^2 G) \circledast I$

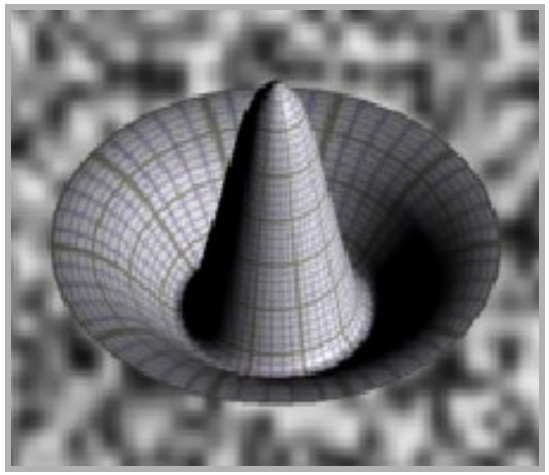
- Thus, can take Laplacian of the Gaussian (LoG) and use that as the operator.

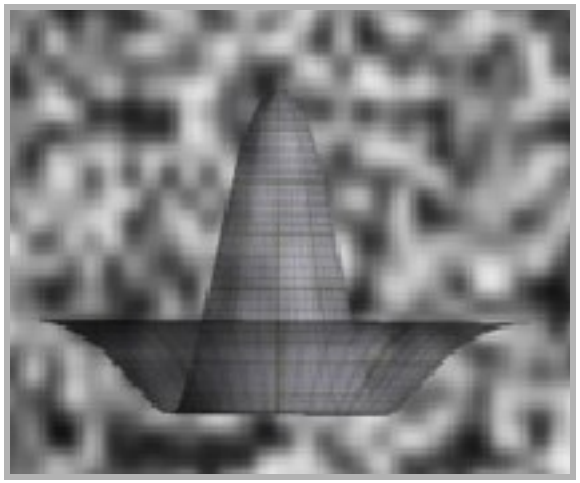
Mexican-Hat Filter

- Laplacian of the Gaussian

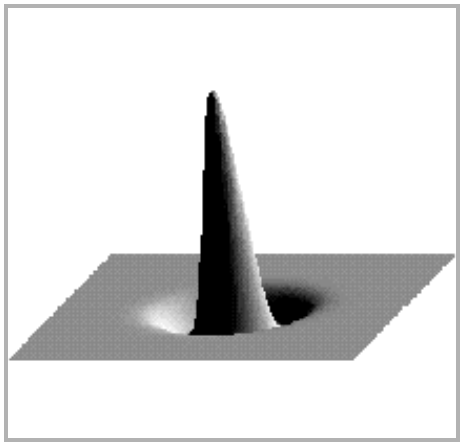
$$\nabla^2 G(x,y) = \frac{-1}{\pi\sigma^4}\left[1 - \frac{(x^2+y^2)}{2\sigma^2}\right]e^{-\left[\frac{(x^2+y^2)}{2\sigma^2}\right]}$$

- $\nabla^2 G$ is a circularly symmetric operator.
- Also called the hat or Mexican-hat operator.
- Sometimes approximated as the difference of 2 Gaussians with different sigma values

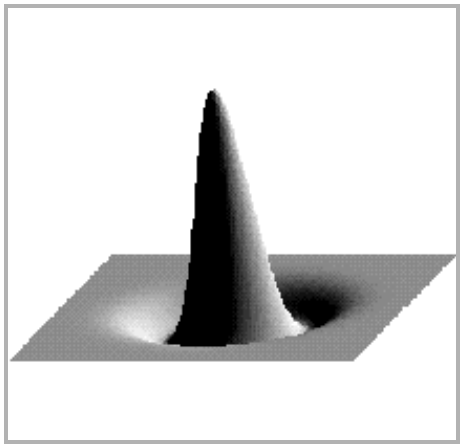




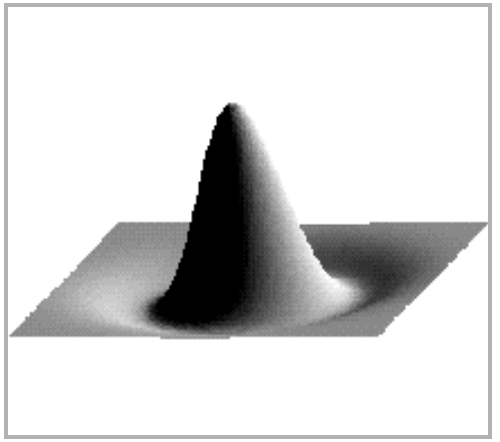
σ^2 Controls Size



$\sigma^2 = 0.5$



$\sigma^2 = 1.0$



$\sigma^2 = 2.0$

Kernels

5x5

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

17 x 17

0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	0	0
0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0
0	0	-1	-1	-1	-2	-3	-3	-3	-3	-3	-2	-1	-1	-1	0	0
0	0	-1	-1	-2	-3	-3	-3	-3	-3	-3	-3	-2	-1	-1	0	0
0	-1	-1	-2	-3	-3	-3	-2	-3	-2	-3	-3	-3	-2	-1	-1	0
0	-1	-2	-3	-3	-3	0	2	4	2	0	-3	-3	-3	-2	-1	0
-1	-1	-3	-3	-3	0	4	10	12	10	4	0	-3	-3	-3	-1	-1
-1	-1	-3	-3	-2	2	10	18	21	18	10	2	-2	-3	-3	-1	-1
-1	-1	-3	-3	-3	4	12	21	24	21	12	4	-3	-3	-3	-1	-1
-1	-1	-3	-3	-2	2	10	18	21	18	10	2	-2	-3	-3	-1	-1
-1	-1	-3	-3	-3	0	4	10	12	10	4	0	-3	-3	-3	-1	-1
0	-1	-2	-3	-3	-3	0	2	4	2	0	-3	-3	-3	-2	-1	0
0	-1	-1	-2	-3	-3	-3	-3	-3	-3	-3	-3	-3	-2	-1	-1	0
0	0	-1	-1	-2	-3	-3	-3	-3	-3	-3	-3	-2	-1	-1	0	0
0	0	-1	-1	-1	-2	-3	-3	-3	-3	-3	-2	-1	-1	-1	0	0
0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0

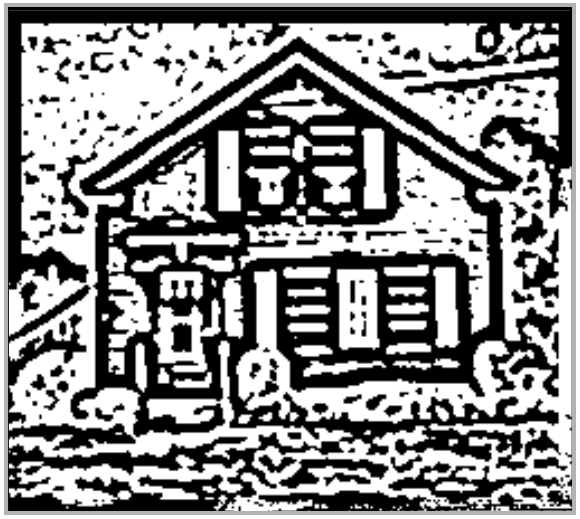
Example

13x13 Kernel

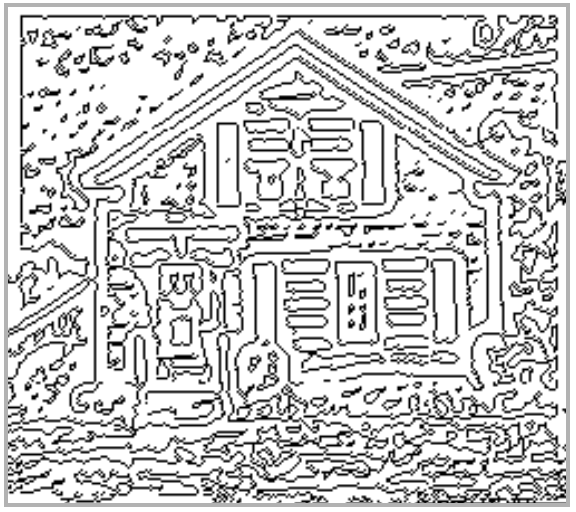
Example

13 x 13 LoG filter

Thesholded Positive



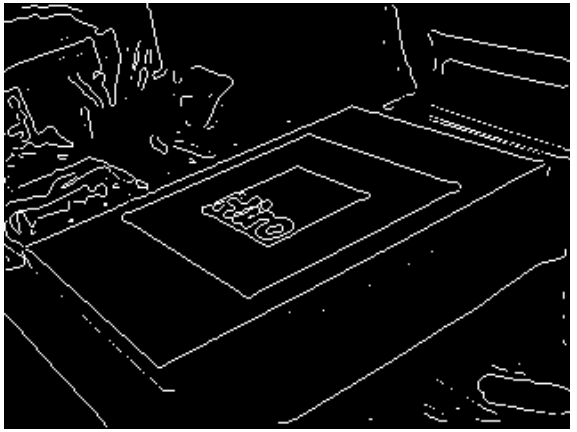
Thesholded Negative



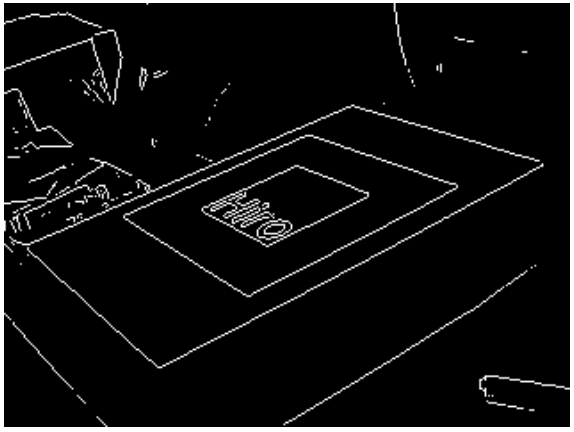
Zero Crossings

Edge Detectors

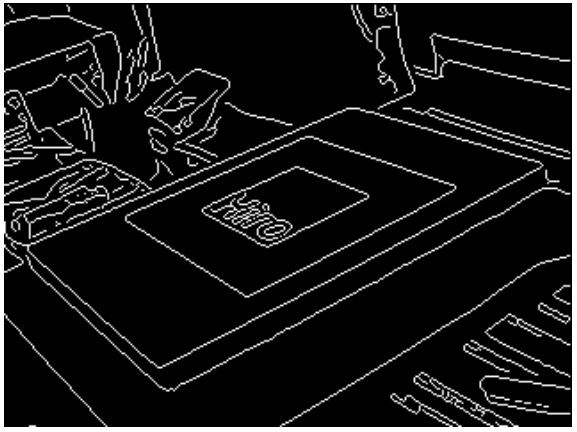
Original image



LoG



Sobel



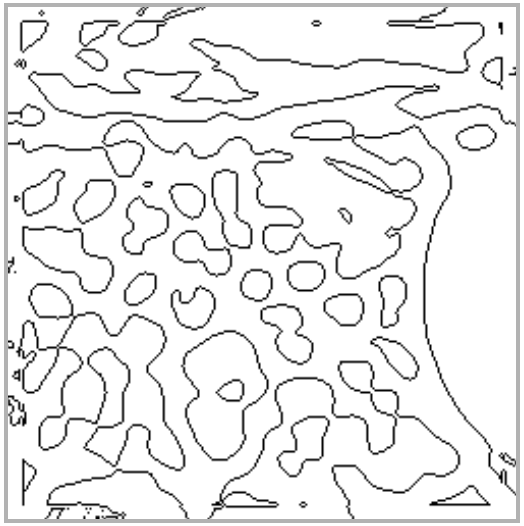
Canny

(detection thresholds automatically chosen by Matlab functions)

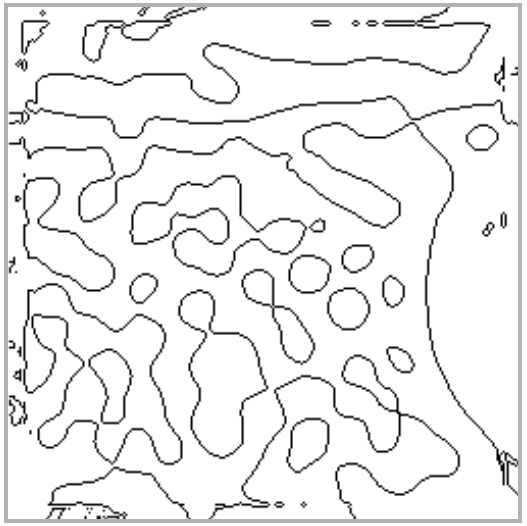
Scale Space

$\sigma^2 = \sqrt{2}$

$\sigma^2 = 2$

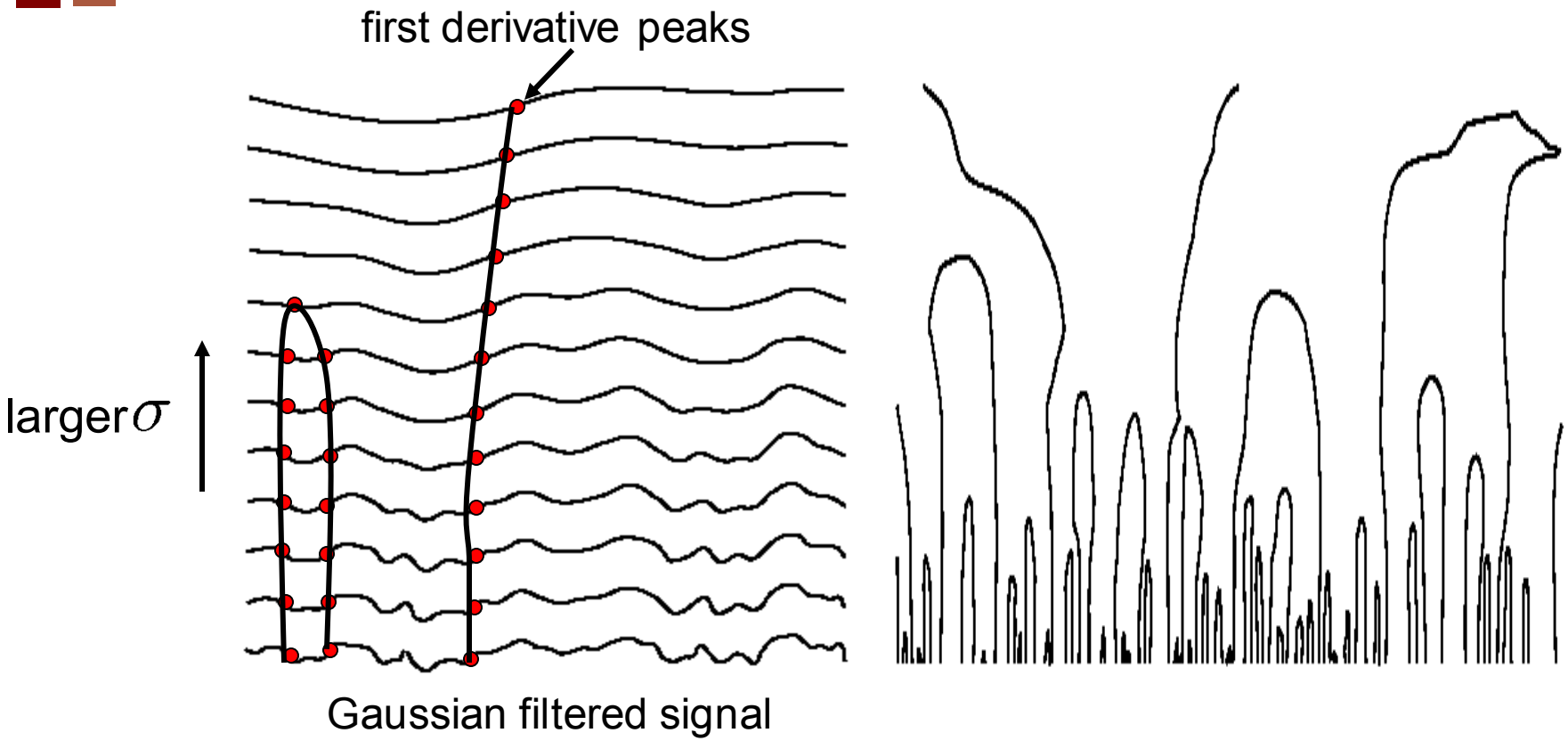


$\sigma^2 = 2\sqrt{2}$



$\sigma^2 = 4$

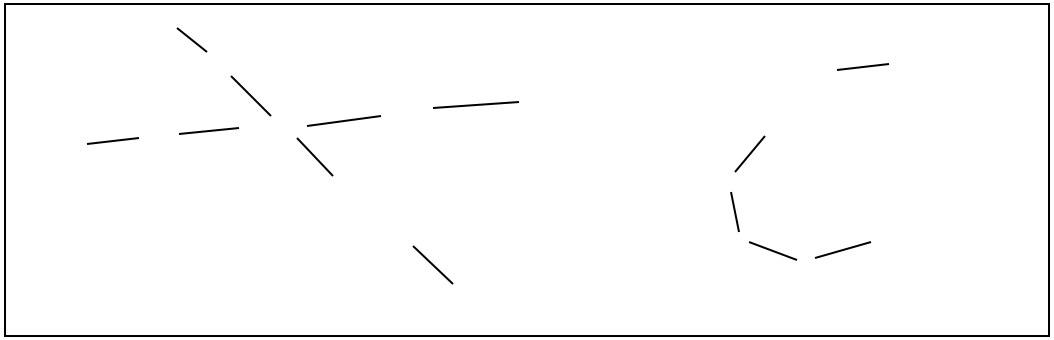
Scale Space



- Properties of scale space (w/ Gaussian smoothing) – Witkin'83
  - edge position may shift with increasing scale (σ)
  - two edges may merge with increasing scale
  - an edge may ***not*** split into two with increasing scale

From Edgels to Lines

- Given local edge elements:



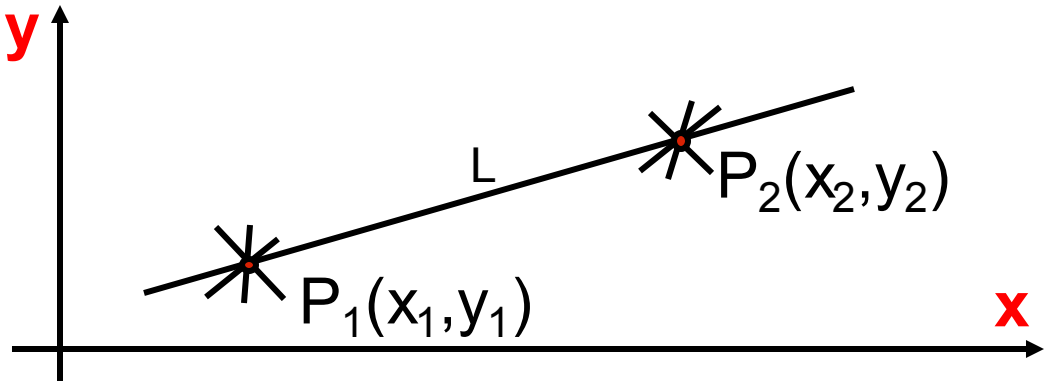
- Can we organize these into more 'complete' structures, such as straight lines?
- Group edge points into lines?
- Consider a fairly simple technique...

Edgels to Lines

- Given a set of local edge elements
  - with or without orientation information
- How can we extract longer straight lines?
- General idea:
  - Find an alternative space in which lines map to points
  - Each edge element 'votes' for the straight line which it may be a part of.
  - Points receiving a high number of votes might correspond to actual straight lines in the image.
- The idea behind the **Hough transform** is that a change in representation converts a point grouping problem into a peak detection problem

Edgels to Lines

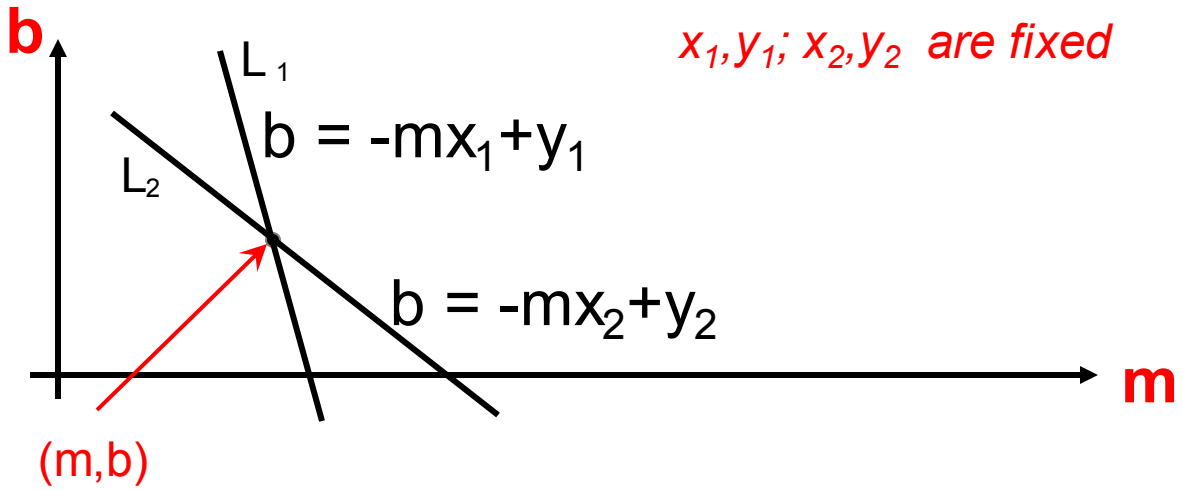
- Consider two (edge) points, P(x,y) and P'(x',y') in image space:



- The set of all lines through P=(x,y) is
  - **y**=m**x** + b,
    for appropriate choices of **m** and **b**.
- But this is also the equation of **a line in (m,b) space**, or <u>parameter space</u>
  - **b** = – x**m** + y

Parameter Space

- The intersection represents the parameters of the equation of a line y=mx+b going through both (x1,y1) and (x2,y2).

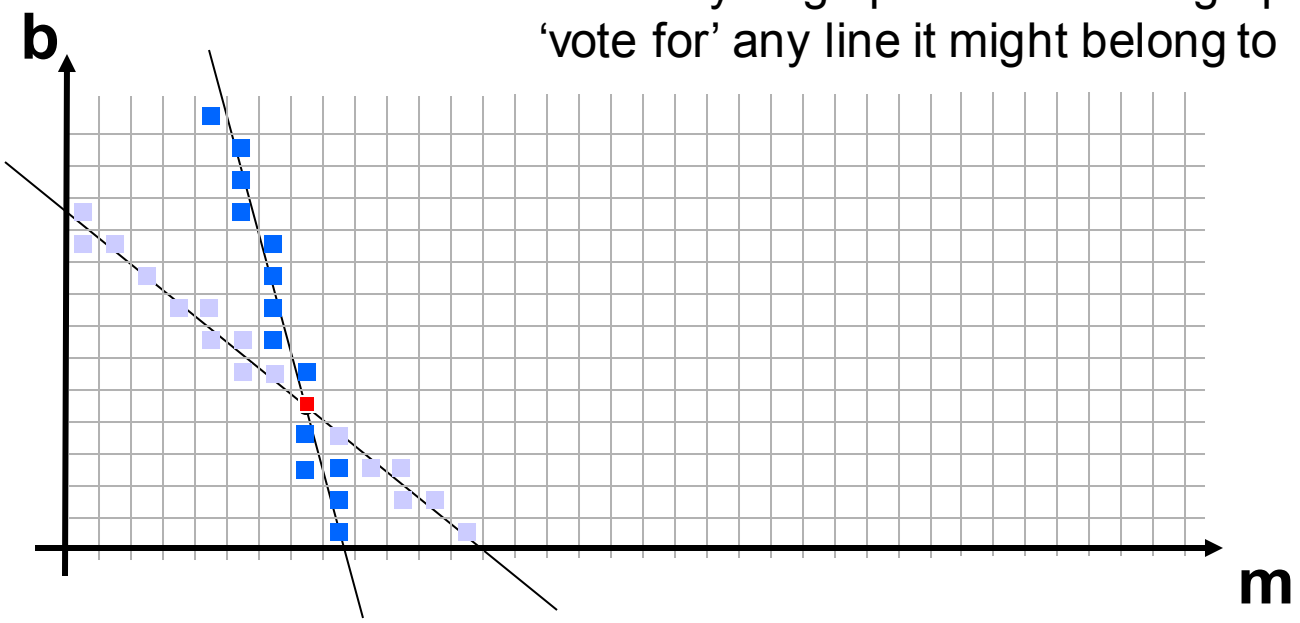


- The more colinear edgels there are in the image, the more lines will intersect in parameter space
- Leads directly to an algorithm (**Hough Transform**)

Quantized Parameter Space

- Quantization

Let every edge point in the image plane
'vote for' any line it might belong to



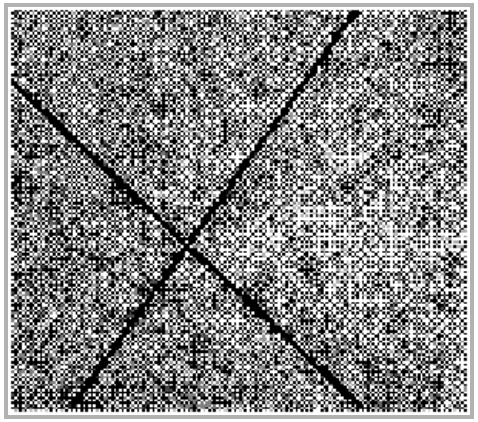
single votes
two votes

The problem of
line detection in image space
has been transformed into the problem of
cluster detection in parameter space

Example

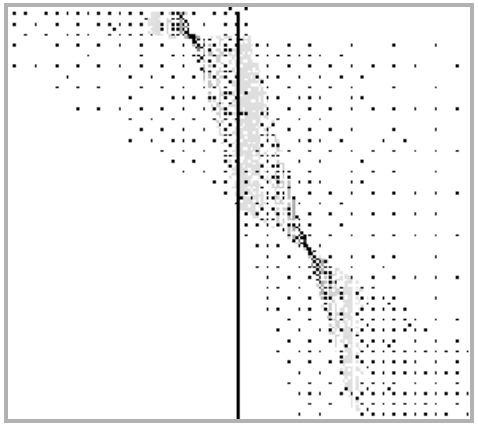
- The problem of line detection in image space has been transformed into the problem of cluster detection in parameter space

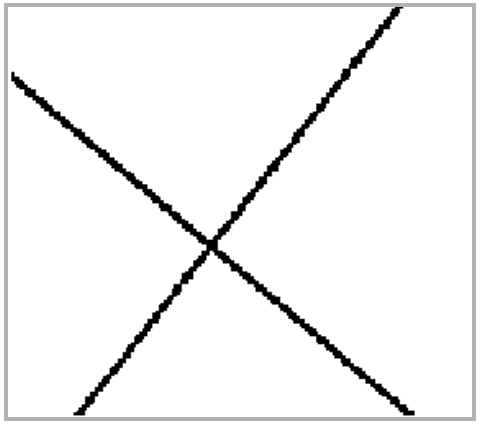
Image



Edges

Accumulator Array



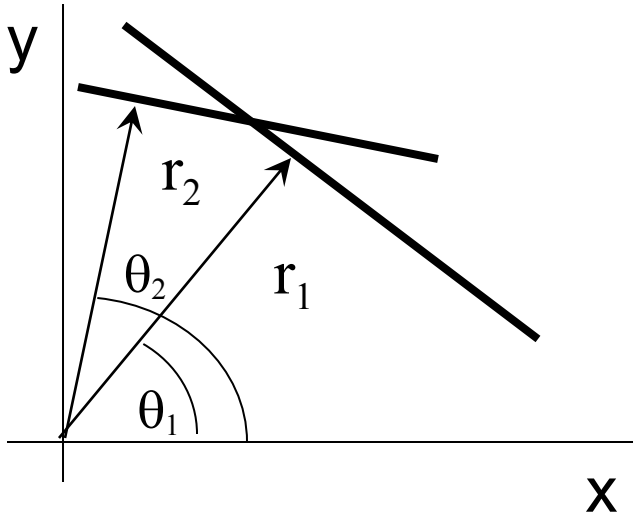


Result

Parameter Space (r,θ)

- Problem: vertical lines have infinite slopes
  - difficult to quantize **m** to take this into account.
- Solution: use alternative parameterization of a **line**
  - polar coordinate representation:

$$\mathbf{r = x \cos \theta + y \sin \theta}$$

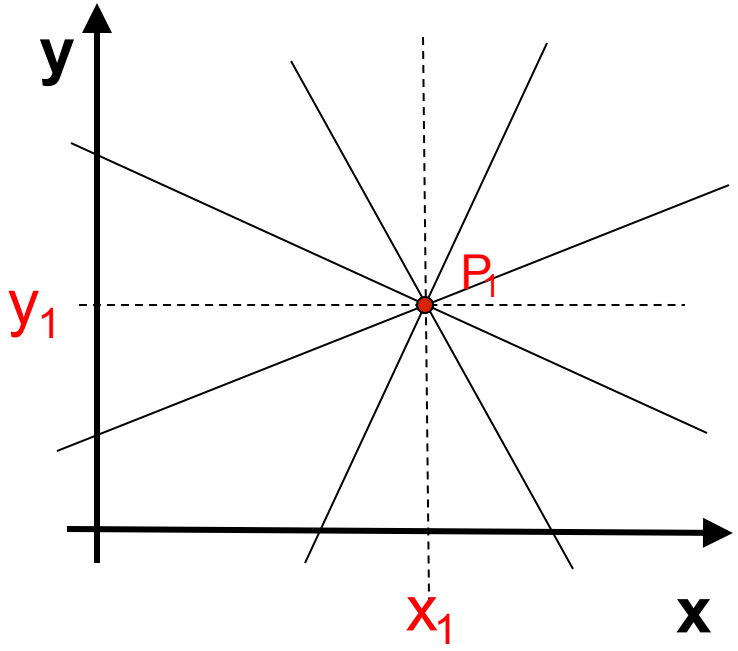


$$r_1 = x \cos \theta_1 + y \sin \theta_1$$

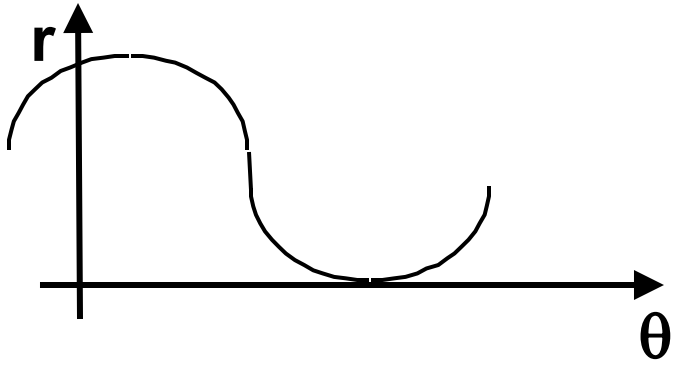
$$r_2 = x \cos \theta_2 + y \sin \theta_2$$

Representation of a point

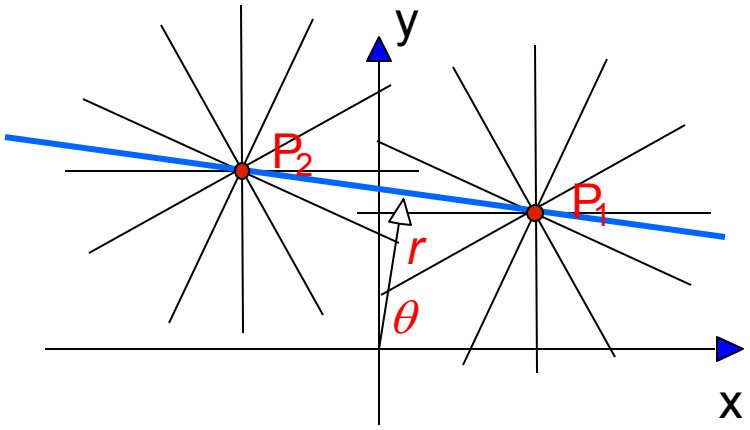
- A point P1=(x1,y1) in image space
  is represented by a sinusoid in (**r**,θ) space



$$r = x_1 \cos\theta + y_1 \sin\theta$$



Example

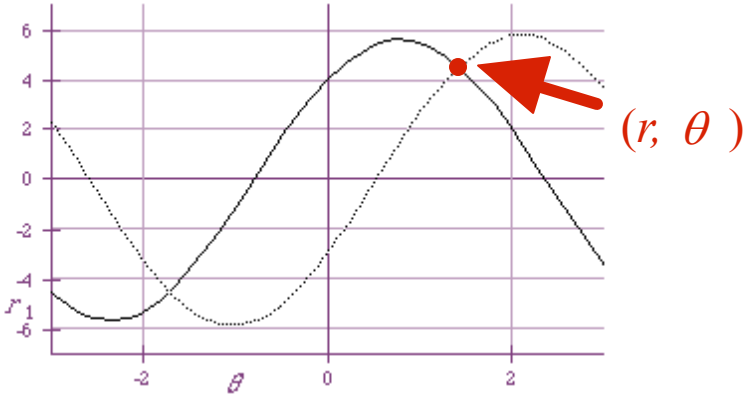


$P_1 = (4,4)$ → $r = x_1 \cos\theta + y_1 \sin\theta =$
$= 4\cos\theta + 4\sin\theta$

$P_2 = (-3,5)$ → $r = x_2 \cos\theta + y_2 \sin\theta =$
$= -3\cos\theta + 5\sin\theta$

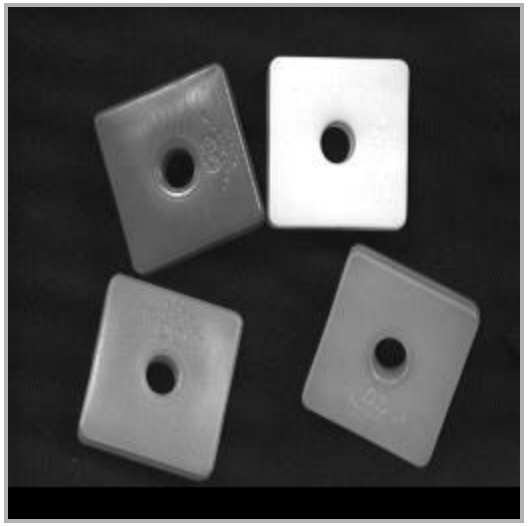
2 eq.s on (r, θ)
Possible to solve for (r, θ)

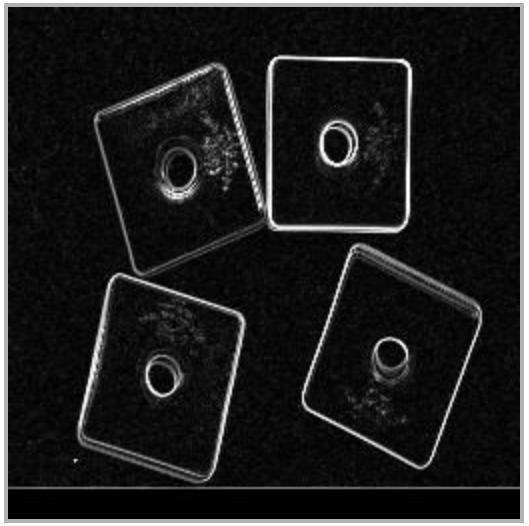
(r, θ) Space



Real Example

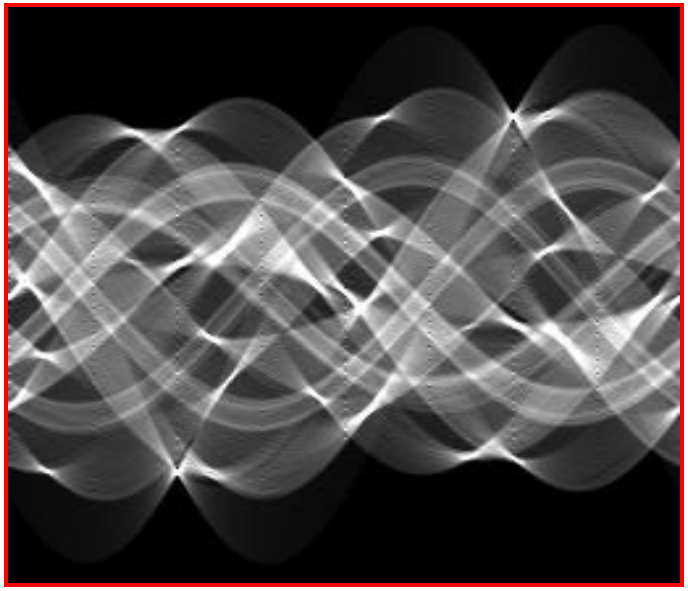
Image

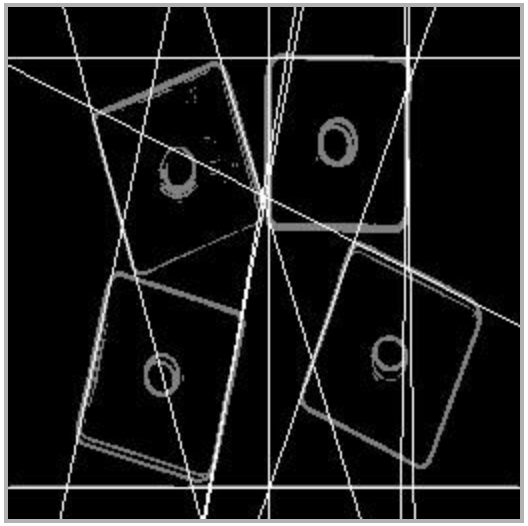




Edges

Accumulator Array

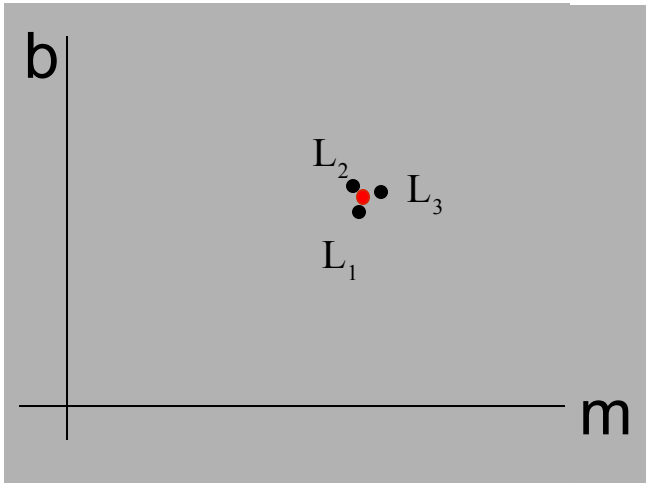




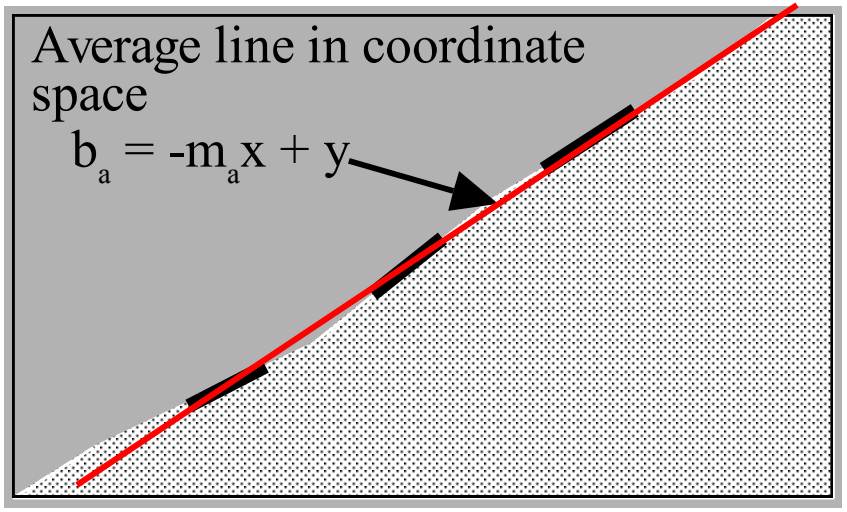
Result

Gradient Data

- ‘Average’ point in Hough Space:

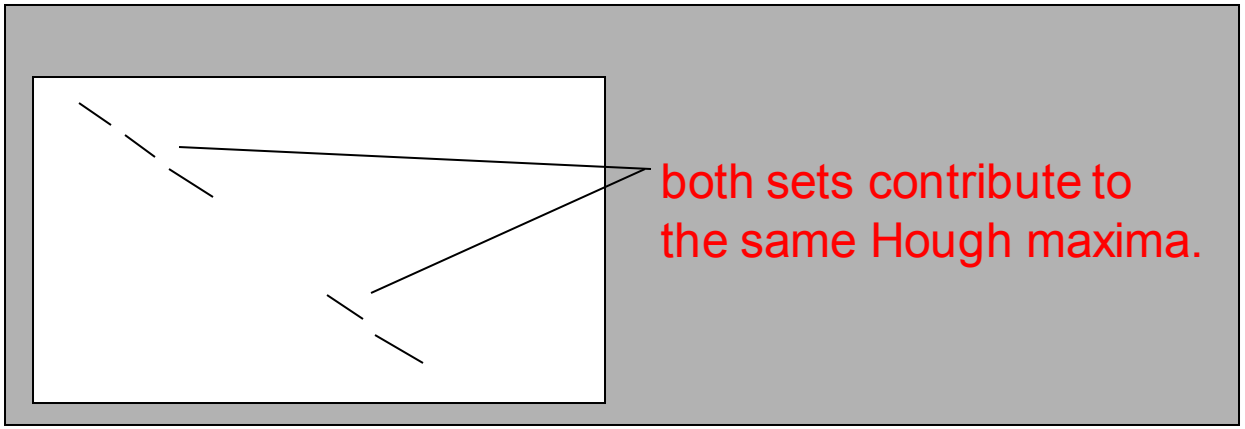


- Leads to an ‘average’ line in image space:



Post Hough

- Image space localization is lost:



- Consequently, we still need to do some image space manipulations, e.g., something like an edge 'connected components' algorithm.

- Heikki Kälviäinen, Petri Hirvonen, L. Xu and Erkki Oja, "Probabilistic and nonprobabilistic Hough Transforms: Overview and Comparisons", *Image and Vision Computing*, Volume 13, Number 4, pp. 239-252, May 1995.

Generalizations

- Hough technique generalizes to any parameterized curve:

$$f(\underline{x,a}) = 0$$

parameter vector (axes in Hough space)

- Success of technique
  depends upon the quantization of the parameters:
  - too coarse: maxima 'pushed' together
  - too fine: peaks less defined
- Note that exponential growth in the dimensions
  of the accumulator array
  with the the number of curve parameters
  restricts its practical application to curves with few parameters

Example: Finding a Circle

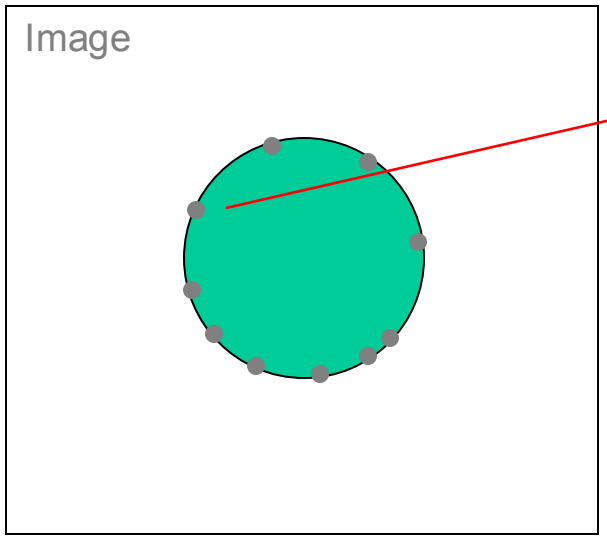
- Circles have three parameters
  - Center: **(a,b)**
  - Radius: **r**
- Circle: $\mathbf{f(x,y,r) = (x-a)^2+(y-b)^2-r^2 = 0}$
- Task:

> Find the center of a circle with known radius **r**
> given an edge image
> with no gradient direction information
> (edge location only)

- Given an edge point at (x,y) in the image, where could the center of the circle be?

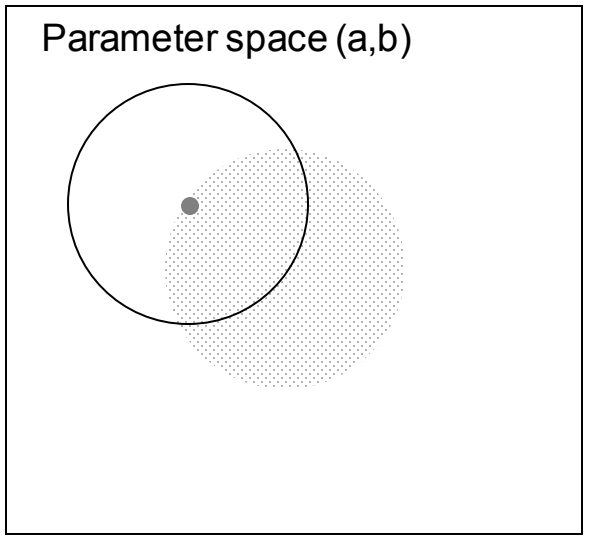
Finding a Circle

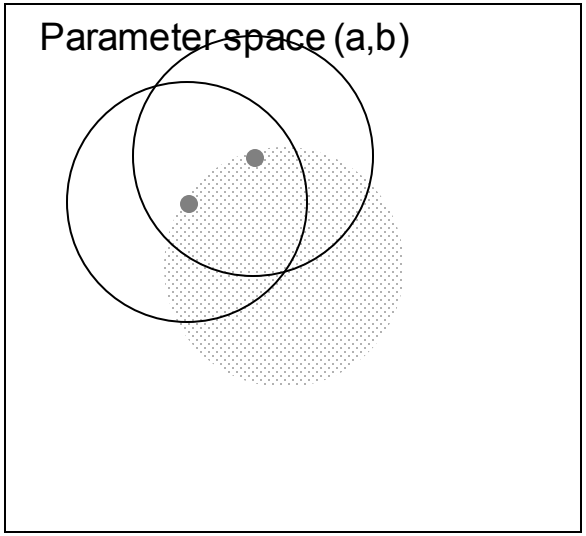
- If radius is known



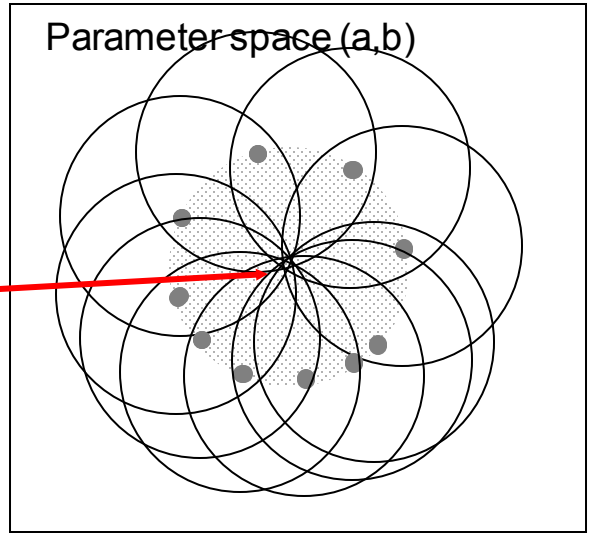
fixed (i,j)

$$(i-a)^2+(j-b)^2-r^2 = 0$$





Circle Center
(lots of votes!)



Finding a Circle

- If we don't know **r**, accumulator array is 3-dimensional

- Hough can be extended in many ways….
  see, for example:
  - Ballard, D. H.
    Generalizing the Hough Transform to Detect Arbitrary Shapes,
    Pattern Recognition 13:111-122, 1981.
  - Illingworth, J. and J. Kittler,
    Survey of the Hough Transform,
    CVGIP, 44(1):87-116, 1988

Hough Transform - Summary

- Hough Transform is a "voting" scheme
  - points vote for a set of parameters describing a line or curve.
- The more votes for a particular set
  - the more evidence that the corresponding curve is present in the image.
- Can detect MULTIPLE curves in one shot.
- Computational cost increases with the number of parameters describing the curve.